Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

A machine learning approach to automatic stroke segmentation [☆]

James Herold [a], Thomas F. Stahovich [b,*]

[a] Department of Computer Science and Engineering, University of California, Riverside, CA 92521, USA
[b] Department of Mechanical Engineering, University of California, Riverside, CA 92521, USA

ARTICLE INFO

Article history:
Received 4 April 2013
Received in revised form
11 September 2013
Accepted 2 October 2013
Available online 5 November 2013

Keywords:
Pen stroke segmentation
Sketch understanding
Pen-based user interfaces
Machine learning

ABSTRACT

We present ClassySeg, a technique for segmenting hand-drawn pen strokes into lines and arcs. ClassySeg employs machine learning techniques to infer the segmentation intended by the drawer. The technique begins by identifying a set of candidate segment windows, each comprising a curvature maximum and its neighboring points. Features are computed for each point in each window based on curvature and other geometric properties. Most of these features are adapted from numerous prior segmentation approaches, effectively combining their strengths. These features are used to train a statistical classifier to identify which candidate windows contain true segment points. ClassySeg is more accurate than previous techniques for both user-independent and user-optimized training conditions. More importantly, ClassySeg represents a movement away from prior, heuristic-based approaches, toward a more general and extensible technique.

1. Introduction

Automatic pen stroke segmentation is the process by which a digital pen stroke is segmented into its constituent lines and arcs. Here, a pen stroke is defined as a continuous sequence of points, which can include sudden changes in direction, so long as the pen remains in contact with the paper. For example, stroke segmentation would decompose a hand-drawn triangle into the three straight lines that comprise it. The challenge in this process lies in determining which bumps and bends in the stroke are intended segment points and which are not. It has been shown that curvature information alone is an unreliable indicator of segmentation [20], and thus a more sophisticated approach is required.

Stroke segmentation is an essential first step in shape recognition [14,16] and thus is a crucial part of sketch-based interfaces. Decomposing a stroke into its constituent geometric primitives also facilitates beautification, in which the hand-drawn primitives are replaced by mathematically precise shapes to produce a neater final result [11,9]. Additionally, as we demonstrate in this paper, stroke segmentation can assist in the automatic semantic labeling of digital pen strokes.

Existing segmentation techniques typically rely on heuristic algorithms and empirically determined parameters [1,8,26,28,25,20,29]. ClassySeg provides greater extensibility and generality than these previous methods by employing general machine learning techniques to identify the segment points in a stroke. ClassySeg begins by identifying a set of candidate windows, each comprising a curvature maximum and its neighboring points. Next, a variety of features, including those that describe curvature, are computed at each point in each candidate window. Most features are taken from previous segmentation approaches, effectively combining their strengths. The features are used to train a statistical classifier to identify which candidate windows contain true segment points and which do not. Lastly, a simple post-prediction step merges overlapping windows that are positively identified by the classifier. We evaluated ClassySeg on a large data set of pen strokes from [8] and found that it is more accurate than previous techniques. Just as important, ClassySeg can be easily extended to include other features and is highly tunable. For example, it can be optimized for different kinds of shapes and can be tuned for individual users and various sketching hardware. ClassySeg currently does not account for interspersed or overtraced pen strokes.

The next section places our approach in the context of previous work. Next, the data set used to evaluate ClassySeg and benchmark it against prior techniques is described. This is followed by a discussion of the main components of ClassySeg, including resampling, candidate window selection, feature computation, the classifier, and the final prediction merging process. ClassySeg's accuracy is then compared to that of three previous segmentation approaches, and to a baseline, naïve segmentation approach. Lastly, we show a novel application of stroke segmentation in which ClassySeg's output is used to semantically label digital pen strokes.

2. Related work

Pen stroke segmentation is a well-researched topic, and numerous methods have been developed. Yu and Cai's [29] technique first attempts to fit a stroke with a single primitive.

[☆] This article was recommended for publication by Beryl Plimmer.
[*] Corresponding author.
E-mail address: stahov@engr.ucr.edu (T.F. Stahovich).

0097-8493/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cag.2013.10.005

If the fit is poor, the stroke is segmented at the point of highest curvature, and the two resulting pieces are recursively processed. Segments are merged in a post-processing phase, but the criteria for doing this are not specified.

The technique of Sezgin et al. [20], which we call SSD, uses speed and curvature to segment pen strokes. Segment points are located at points of minimum speed and maximum curvature. This work demonstrates the usefulness of pen speed data for segmentation, and shows that curvature data alone is inadequate. SSD is suitable for segmenting pen strokes into sequences of line segments, but cannot handle arcs.

Wolin et al. [25] developed ShortStraw, which begins by resampling the pen stroke, and then computes the “straw value” for each point (see Section 4.3.3), which gives an indication of the local curvature. All points with a straw value below an empirically determined threshold are considered candidate segment points. A top-down phase then examines the segments between each pair of consecutive candidate points to evaluate the quality of the line fit. If the fit is poor, segment points are added.

Xiong and La Viola [28] developed iStraw, which improves upon the ShortStraw approach by including timing information and curve detection. iStraw achieves better accuracy than ShortStraw and is able to handle curve and arc segments, which ShortStraw cannot.

The technique of Wolin et al. [26], called Sort, Merge, Repeat (SMR), begins just as SSD does, by locating candidate segment points at speed minima and curvature maxima. The algorithm then finds the shortest segment and merges it with one of its neighboring segments in an attempt to remove false positives. This process is repeated until the error in fitting lines and arcs to the segments is below an empirically determined threshold.

Recently, Herold and Stahovich [8] presented SpeedSeg. This approach also identifies the initial candidate segment points at speed minima and curvature maxima. A set of heuristics are then used to both merge and split the initial segmentation to produce a more accurate final result. The heuristics consider several geometric and speed-based features with empirical thresholds. These thresholds can be optimized to improve performance.

Albert et al. [1] have presented a method called Tangent and Corner Vertices Detection (TCVD), which uses the radius function of a pen stroke to identify segment points. The radius function the authors use is less susceptible to noise than the traditional curvature computation used in prior work, yet this method still relies upon identifying minima in the radius function, which is similar to other prior techniques.

Nearly all of these approaches rely on heuristics and empirical parameters, which limits their extensibility. In many cases, there is no automated procedure for selecting optimal parameter values. By contrast, ClassySeg uses a general-purpose machine learning approach that naturally extends to incorporate any number of features. Here, we use a collection of features derived from multiple, existing segmentation techniques, but other features can be directly added. Furthermore, ClassySeg is highly optimizable, as it automatically identifies optimal parameter values via a trained classifier. As a result, ClassySeg can be easily tuned for specific users, specific classes of shapes, and specific sketching hardware.

This technique is similar to other sketch processing techniques that employ machine learning. For example, Peterson et al. [17] use a classifier based approach for stroke grouping. The goal of that work is to group pen strokes from a freely drawn sketch into clusters representing individual objects. The authors accomplish this by first computing features that capture spatial and temporal properties of the pen strokes and the spatial relationships between them. A statistical classifier then uses these features to classify the strokes according to their type. A second classifier then examines pairs of strokes of the same type to determine if they should be clustered together into an object. Similarly, Blagojevic et al. [3] have applied data mining techniques to separate text and shape pen strokes in digital ink diagrams. In particular, they compute a variety of features characterizing the spatial and temporal properties of the pen strokes and use classifiers to distinguish text from shapes. They evaluated the performance of several classifiers, such as LADTree and LogitBoost, for this task. Both of these research efforts demonstrate a movement toward machine learning based sketch processing techniques and show that these techniques can be more effective than prior approaches.

3. Data set

We used the pen stroke data from the study described in [8] to evaluate ClassySeg's performance and benchmark it against prior segmentation methods. In that study, an HP TC4400 Tablet PC with a digitizer resolution of 1024×768 pixels was used for data collection. Fourteen subjects were asked to draw each of the ten symbols from Fig. 1 six times.

Subjects were informed that the purpose of the study was to collect data to evaluate the performance of an algorithm. Subjects were instructed to “draw naturally with ordinary accuracy,” and to not attempt to “trick or break the system.” Before beginning the exercise, each subject was given a few minutes to practice sketching on the Tablet PC.

Each point from the data set is a triple containing an x-coordinate, y-coordinate, and time value. To facilitate training and testing of algorithms, the true segment points for each pen stroke were manually labeled using an approach analogous to that in [27]. Specifically, the pen stroke data was initially segmented using the technique described in [21]. Then, segment points were manually added, deleted, and moved as necessary to achieve the correct segmentation.

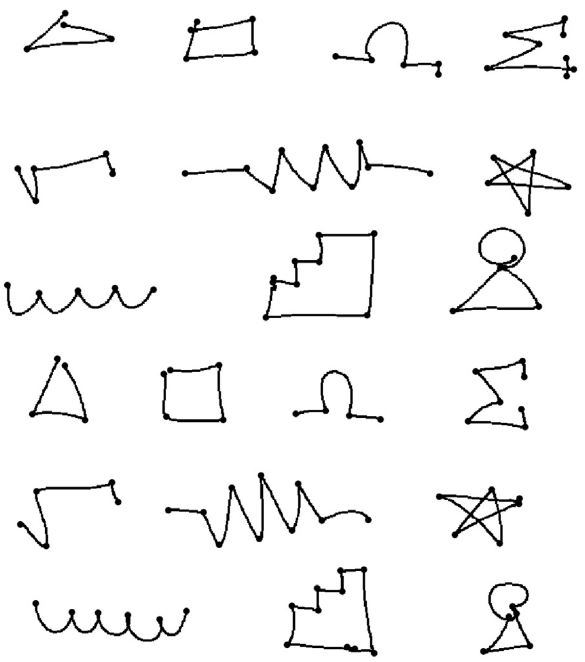

Fig. 1. The ten shapes used in the user study: triangle, square, omega, sigma, square root, spring, star, wave, stepped-block, and pivot. The top ten examples are from one subject, the remaining ten are from another. Large data points denote manually labeled segment points. Note that the number of segments comprising a shape may differ from one subject to the next.